FINAL YEAR PROJECT DEFENSE

LensRAG

Modality-Preserving Video Retrieval with User-Defined Analytical Lenses

Sanjay Reddy Choudapur

Department of Computer Science · Georgia State University

Schoudapur1@student.gsu.edu

Spring 2026

Why is long-video QA still hard?

Two failure modes in current systems

Failure Mode 1: Frame-budget ceiling

Large Video-Language Models (LVLMs)

hit a fixed frame budget well before a long clip ends.

- 1-hour video ≈ ~7-10k frames + thousands of spoken words
- Context window forces aggressive temporal downsample
- Accuracy is non-monotonic past a threshold — more frames crowd out the ones that mattered.

Failure Mode 2: Premature flattening

Most Video-RAG systems collapse every modality

into one text blob before retrieval runs.

- ASR + OCR + detector outputs → single text index
- Whatever the fixed extractors did not transcribe is lost
- New analytical question ⇒ rewrite the extractor
- Visual semantics beyond the extractor's skills are invisible

Goal: keep the modalities separate until after retrieval, and let the user's analytical angle be a first-class input.

Related work, and where LensRAG differs

Three families of prior work

LVLMs (end-to-end)

Video-ChatGPT, VideoChat,
Video-LLaVA, LongVA, LongLLaVA

- Fixed frame budget
- Accuracy plateaus / regresses past a point
- Long clips force aggressive downsampling

LensRAG's answer:

Keep raw frames as a first-class index (CLIP text→image retrieval)

Video-RAG systems

Video-RAG, iRAG, DrVideo,
VideoTree, MovieChat, LLoVi

- Extractors picked up-front (ASR/OCR/detector)
- Single flattened text index after extraction
- Hard to adapt to new analytical angles

LensRAG's answer:

Keep each modality in its own embedding space; fuse only at scoring

Agentic planners

VideoAgent (ECCV '24),
ReAct-style tool-calling planners

- Strong but very slow
- High $ per query
- Hard to reason about end-to-end for an interactive UI

LensRAG's answer:

Inference is a single parallel retrieval + one generator call — cheap and fast

LensRAG: four parallel indexes + a user lens

Mix the modalities only at the scoring step

Visual Index (CLIP)

Text→image similarity over sampled frames

Audio Index (Whisper + embed)

Semantic search over 10 s transcript chunks

Description Index (Gemini 2.0 Flash + embed)

Per-frame visual captions, generic

Lens Index (lazy, per user intent)

Per-frame captions written through the user's analytical angle

Retrieval → Fusion → Generation → Grounding

1) 4 parallel `.similarity()` calls, Top-K = 3 per index.

2) Min-max normalize each index's scores into [0, 1].

3) Weighted sum over the four legs:

`weights w = (0.30, 0.20, 0.30, 0.20)`

over (audio, image, desc, lens).

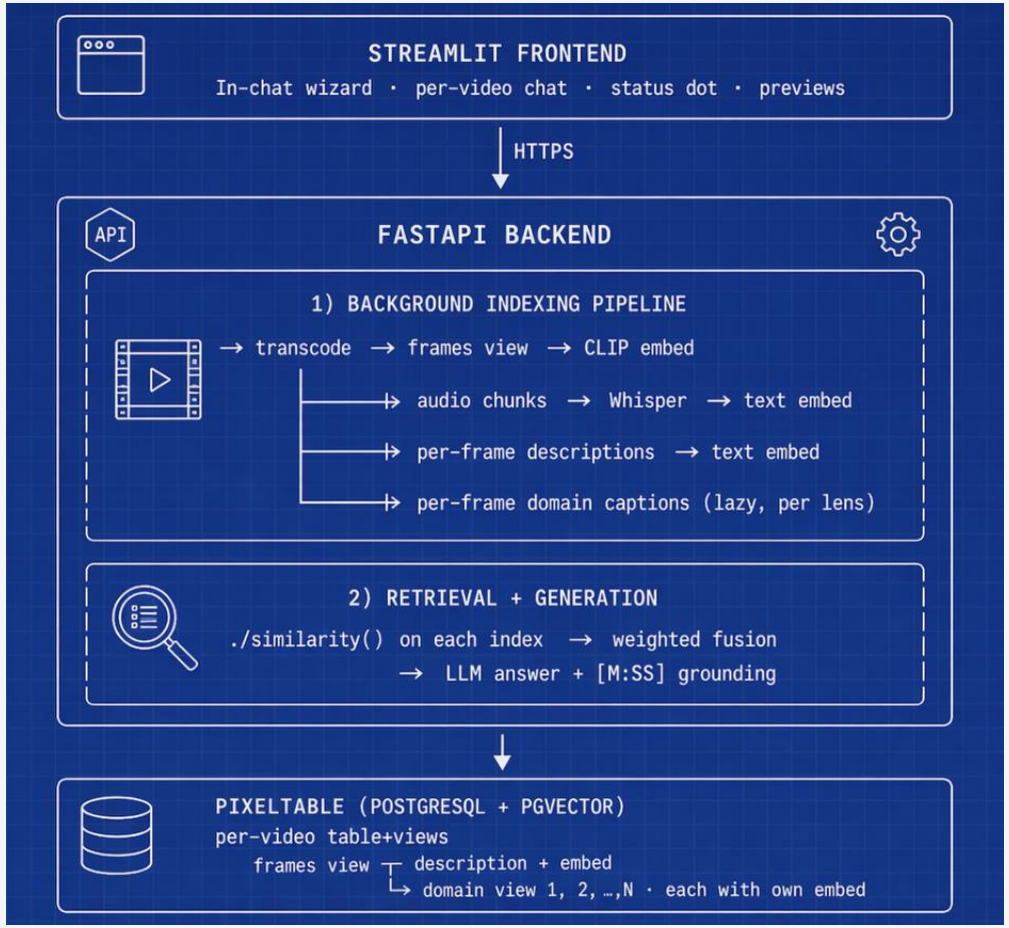
4) Dedup within a ±1 s window, keep Top-N = 10.

5) Generator (Llama 3.3 70B) writes a `[M:SS]`-cited answer.

6) Grounding filter: keep citations within ±3 s of an emitted timestamp; otherwise flag **Ungrounded**.

Deployment stack

Streamlit UI → FastAPI backend → Pixeltable (pgvector)



What's on each layer

Frontend - Streamlit Cloud

- In-chat wizard (upload → lens pick → indexing progress)
- `Auto-polling via @st.fragment(run_every=5)`
- Per-video chat sessions; inline [M:SS] chips; Ungrounded badge

Backend - FastAPI on Railway

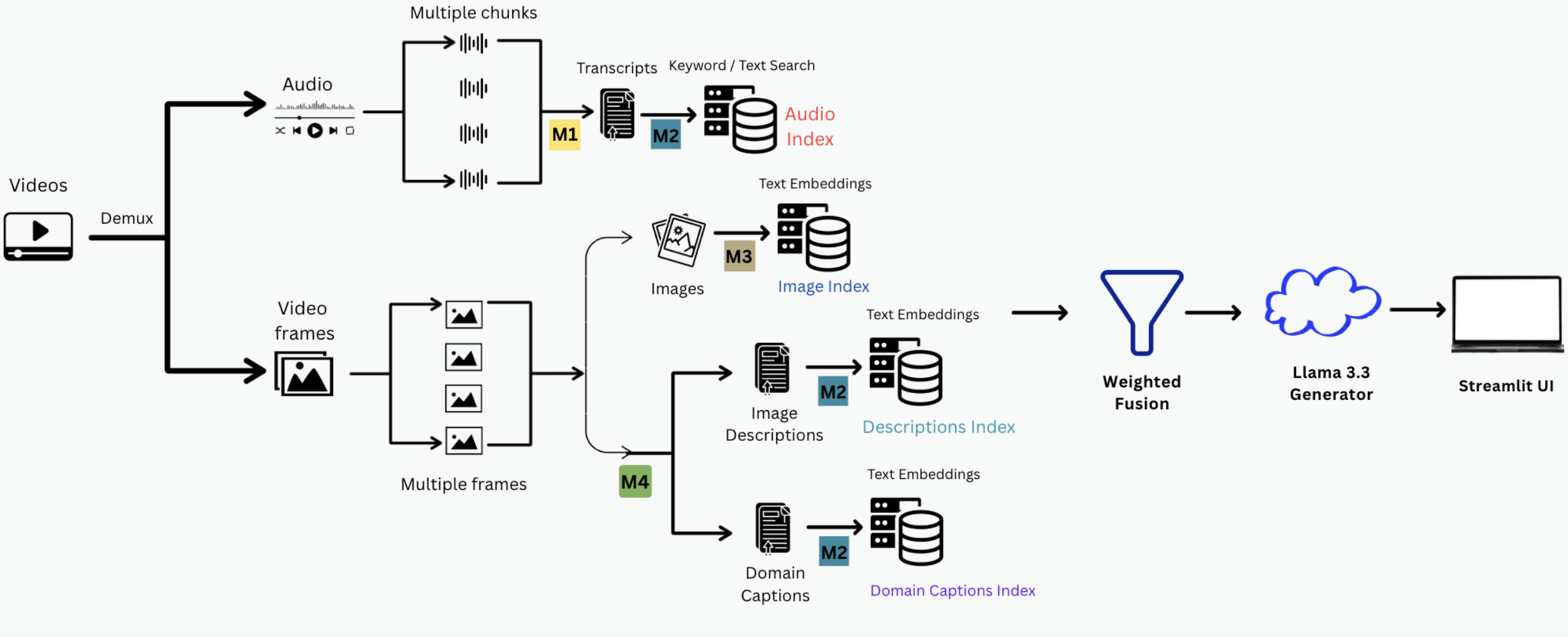
- Background daemon thread per upload → isolated asyncio loop
- `ProcessingStateStore (RLock + disk snapshot) survives restarts`
- H.264 High → Main transcode inline; 500 MB / 2 h cap

Data - Pixeltable (PostgreSQL + pgvector)

- 1 table per video + frames / audio / description / lens views
- Up to 5 lens views per video, LRU-evicted
- Embedding indexes on every semantic column

Indexing pipeline at ingest time

Four parallel legs build on one shared frames view



All four legs run async. Adaptive frame sampling (40 / 60 / 80 / 100 / up to 150) keeps indexing cost sub-linear in duration.

The lens, made question-informed

Real users set a lens before asking. We simulate that.

Pipeline: questions → lens phrase → per-frame captions

1) Collect the user's planned questions (or the demo inputs).

2) Summarize them into a single 6-14 word lens phrase via the chat LLM.

3) Hash with blake2b → view name `{video}_domain_{hash8}`

4) Each frame is re-captioned under a structured prompt that always covers:

(a) lens-focused content,
(b) on-screen text / numbers / logos verbatim,
(c) people + objects with counts and positions.

5) LRU-evict at 5 lens views / video; lazy create on first use.

Before vs. after (real examples)

Sports clip

Sub-category lens: *"sports and athletic performance"*
Question-informed: *"identifying players their actions and on-screen jersey numbers"*

Documentary clip

Sub-category lens: *"historical events narrative"*
Question-informed: *"historical events and on-screen date or place labels"*

Cooking clip

Sub-category lens: *"cooking technique"*
Question-informed: *"cooking ingredients and utensils with visible colors and food textures"*

The new lens explicitly names OCR-sensitive cues (numbers, labels, dates, counts) when questions need them.

System implementation details

What's in the repo, pinned versions

Language + runtime

- Python 3.11.9 (backend/runtime.txt)
- FastAPI 0.115+ · Uvicorn
- Streamlit 1.40+ for the UI

Vector store

- Pixeltable 0.5.x
- PostgreSQL + pgvector under the hood
- Per-video table + 4-5 views + embed index on every

ML + media

- PyTorch 2.1.x CPU-only
- CLIP ViT-B/32 run locally (512-d vectors)
- FFmpeg via subprocess; moviepy / Pillow

Embedding dims

- CLIP frame → 512-d (image space)
- text-embedding-3-small → 1536-d
- Audio / description / lens: all 1536-d text vectors

Retrieval knobs

- Top-K per index = 3, Fused Top-N = 10
- Temporal dedup window = 2 s
- Citation grounding tolerance = ±3 s

Hardware

- GPU: none required at serving time
- CPU: Railway default container profile
- Memory: ≥ 4 GB steady, +few 100 MB per ingest

Prompt templates

Answer gen: modality-labelled context blocks + cite timestamps as [M:SS].

Description: 1-2 sentence caption; no speculation beyond the frame.

Domain (lens): 3-4 sentences covering lens angle + on-screen text verbatim + counts and positions.

Evaluation setup

Stratified Video-MME slice + 5-way ablation

Dataset & protocol

Benchmark: Video-MME (multi-modal long-video QA, Fu et al. 2024)

Slice: stratified 30 videos = 10 short + 10 medium + 10 long
→ 90 multiple-choice items (3 per video)
→ 4.5 h of video processed end-to-end

Metric: Top-1 accuracy on the MC option (A/B/C/D),
plus grounded rate and per-task breakdown.

Determinism: fixed seeds 42 + 43 for sampling;
idempotent per (config, question_id) eval harness.

5 configurations — additive ablation

Configuration	w_aud	w_img	w_desc	w_lens
Audio only	1.00	0.00	0.00	0.00
Image only	0.00	1.00	0.00	0.00
Audio + Image	0.60	0.40	0.00	0.00
+ Description	0.35	0.25	0.40	0.00
Full LensRAG	0.30	0.20	0.30	0.20

All 5 configs run over the same 90 QA items. Weight-patching is asserted inside the harness (a fresh ResultFusion() is built per config, and captured weights are checked against the target vector).

Results

Every added index lifts accuracy. Monotonic.

Primary comparison (n = 90 QA / config)

System	Top-1 Acc.	Grounded rate
Audio-only retriever	35.56 %	11.11 %
Image-only retriever	38.89 %	22.22 %
LensRAG (full)	62.22 %	8.89 %

Headline

+26.66 %

LensRAG over the strongest single-modality baseline.

95 % bootstrap CI [52.2 % – 72.2 %].

Additive ablation (each row: was this index worth it?)

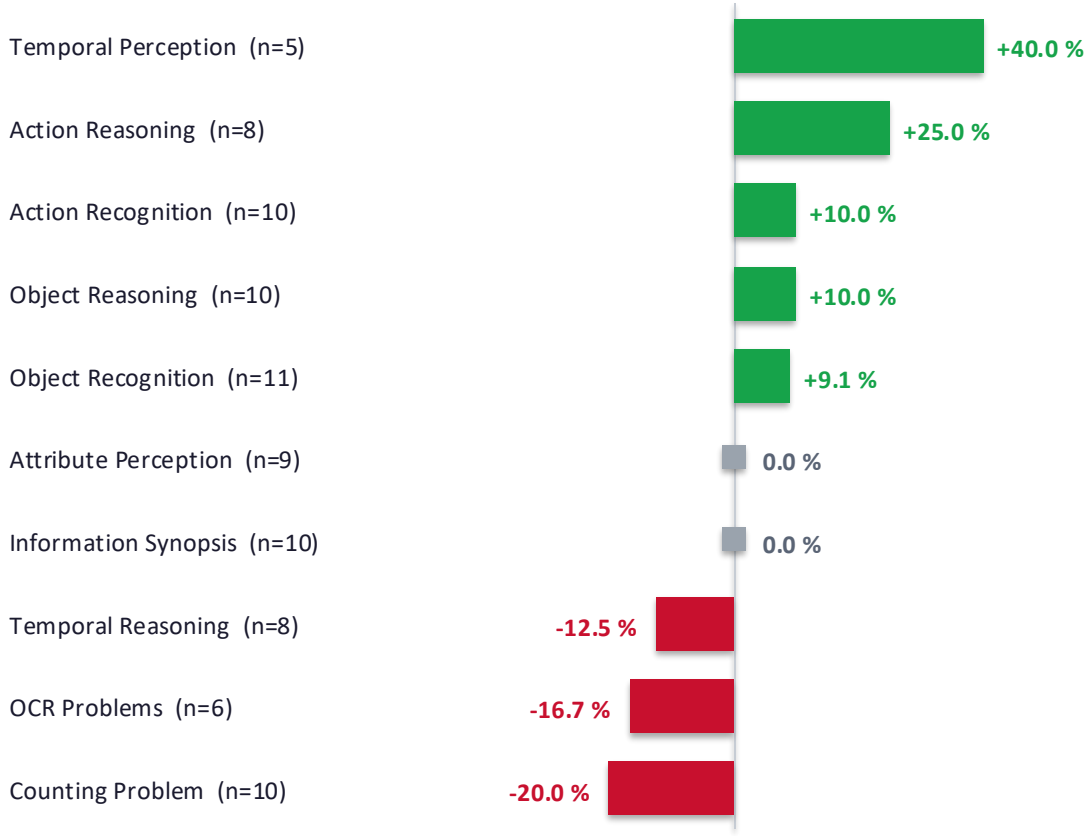
Active indexes	Top-1	Δ vs. prev.
Audio only	35.56 %	—
Image only	38.89 %	+3.33 %
Audio + Image	41.11 %	+2.22 %
+ Description	58.89 %	+17.78 %
+ Lens (full)	62.22 %	+3.33 %

Every step is positive, including +Lens. The description index carries the biggest single gain (+17.78 %); the lens adds further lift on top, so each of the four indexes contributes independent signal.

Where the lens specifically helps

Per-task and per-duration breakdown — n = 90

Full minus +Description, per Video-MME task



Per duration bucket (Top-1, %)

Bucket	+Desc	Full	Δ
Short (n = 30)	60.0	60.0	0.0
Medium (n = 30)	60.0	60.0	0.0
Long (n = 30)	56.7	66.7	+10.0

Takeaway

· The lens helps most on **action-oriented, temporal, and object-identification** questions.

· The lens wins cleanly on **long clips (+10 pts)** where richer captions matter more.

· Small losses on OCR / Counting / Temporal Reasoning are where the lens de-prioritises exact counts or exact ordering. They don't swamp the aggregate gain.

Qualitative case studies

One archetype per deciding index — real Video-MME items

1 Dialogue-heavy · deciding: Audio

Item 238-1

Q: Which of the following best describes the topic of the video?

Image-only baseline → C (incorrect)

LensRAG → D (correct)

Why LensRAG wins

Top-3 fused sources: audio, audio, audio.
The transcript chunk contained the topic statement verbatim.

2 Purely visual · deciding: Image

Item 183-3

Q: Which best describes the audience's reaction to the singer's performance?

Audio-only → None (refusal, wrong)

LensRAG → A (correct)

Why LensRAG wins

Top-3 fused sources: image, image, image.
CLIP frame matches captured the crowd's emotional reaction.

3 Lens-dependent · deciding: Lens

Item 234-3

Q: What kind of food is being prepared in the video?

Full, lens disabled → A (wrong)

Full LensRAG → D (correct)

Why LensRAG wins

The cooking lens caption named the specific dish on-screen.
Without the lens, captions were too generic to match.

Efficiency and cost

Training-free, CPU-only, $0.015 per 100 queries

Indexing

~13 min

wall-clock per hour of video
(758 s / hr, all 4 eager indexes)

Query latency

4.07 s

total, cached lens
(retrieve 1.50 s + gen 2.57 s)

Cost / 100 q

$0.0147

dialogue-heavy workload
(Llama 3.3 70B via OpenRouter, Apr '26)

GPU required

none

CPU-only CLIP + remote vision / generator
Railway default container

Indexing wall-clock (measured on 30 videos, 273 min total)

Stage	20 s fixture	Per hour
Transcode + frame view	5 - 10 s	174 s
Audio (Whisper + embed)	5 - 10 s	443 s
Image (CLIP, CPU)	10 - 20 s	109 s
Description (vision + embed)	20 - 30 s	32 s
Lens (lazy, per new lens)	20 - 40 s	29 s
Eager total (before chat)	40 - 70 s	758 s

Per-query latency & cost (N = 50 warm queries per workload)

Stage	Dialogue-heavy	Visual-heavy
Retrieval (4 × .similarity)	1.50 s	1.42 s
Fusion + dedup	0.000 s	0.000 s
Generator answer	2.57 s	1.29 s
Citation grounding	0.000 s	0.000 s

Workload	Tokens / q	Cost / 100 q
Dialogue-heavy	1,009	$ 0.0147
Visual-heavy	825	$ 0.0112

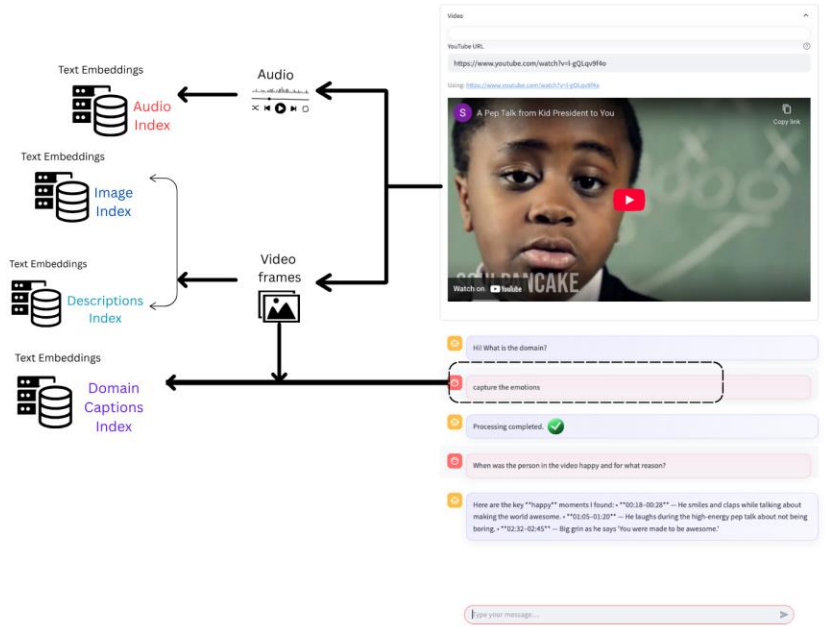
Live demo walkthrough

Upload → pick a lens → ask three archetype questions

Demo script · ~ 3 minutes

0	**Open**	https://video-ragdlproject-hezrklx8rnhbjwucqvegxq.streamlit.app
1	**Upload**	Drop a short Video-MME clip. Watch the in-chat wizard.
2	**Lens**	Pick a preset (Emotions / Marketing / Educational …) or type a custom lens. Confirm.
3	**Wait**	Auto-polling shows per-stage indexing progress. Chat opens when audio / image / description finish.
4	**Dialogue Q**	"What did the speaker say about X?" Expected deciding index: audio
5	**Visual Q**	"What color is the object being shown?" Expected deciding index: description or image
6	**Lens Q**	"From a <lens> angle, what is happening?" Expected deciding index: lens
7	**Inspect**	Hover on the [M:SS] chips → jumps to that timestamp Open the citations panel → see retrieved chunks

The UI at a glance



Four index sidebar + chat panel. Answers show [M:SS] chips; an "Ungrounded" caption shows when no timestamps were cited.

Limitations — where LensRAG still falls short

Honest list with remediation path

OCR not an explicit index

Pixel-only text (signs, slides) is recoverable via the description / lens captions but never as a first-class signal.

Fix: *Add a fifth OCR leg; the Pixeltable recipe applies unchanged.*

Audio ceiling = Whisper-1

Noisy recordings, heavy accents, overlapping speakers degrade transcripts; retrieval over those transcripts degrades in lockstep.

Fix: *Swap in newer ASR; add diarisation.*

Lens prompt sets the lens ceiling

Bland user lenses ("stuff about marketing") yield bland captions. UI presets are written to avoid this; custom lenses inherit user prompt quality.

Fix: *Auto-refine lenses from questions (already done in eval).*

Embedding storage scales w/ frame count

4 + up-to-5 lens indexes per video × (40-150 frames) × 1,536-d vectors. Fine at project scale; not at thousands of videos.

Fix: *Quantise + tier to Postgres cold storage.*

Retrieval errors propagate

Generator only sees retrieved chunks. A missed chunk → hallucination (grounding flags it) or a polite refusal. Grounding catches the first mode, not the second.

Fix: *Hybrid sparse + dense retrieval; reranker.*

First-lens-on-long-clip latency

Lazy lens-view build happens synchronously inside /chat. First query can overshoot Railway's 60 s edge-proxy cap.

Fix: *Move lens build to a background worker.*

Conclusion & future work

Done, and what comes next

What this project delivered

- ✓ A training-free video-RAG system with four semantic indexes + a user-defined lens.
- ✓ +26.66 points over the best single-modality baseline on a stratified 30-video Video-MME slice.
- ✓ Monotonic ablation: audio → +image → +desc → +lens, every step positive, each index matters.
- ✓ Explicit citation grounding + “Ungrounded” UI flag.
- ✓ Deployed end-to-end (Railway + Streamlit Cloud), 117 unit tests + cassette-replayed integration suite.
- ✓ Reproducible eval harness (8 scripts, JSONL audit trail).

Future work

- **Add a dedicated OCR leg**
  The 6 % losses on OCR-style questions point to this directly.
- **Background lens-view builds**
  Decouple first-chat from lens creation → no 60 s proxy cap.
- **Streaming answers through grounding**
  Wire existing generate_streaming_answer into the grounding pass.
- **Query-adaptive fusion weights**
  Learn per-question-type weights rather than the current constant.
- **Wider evaluation**
  Scale to full Video-MME subset (270 QA) and add EgoSchema / MVBench.
- **Open-weights generator**
  Swap Llama 3.3 70B for a self-hosted open model to cut external dep.

Thank you.

Questions, critiques, and deeper dives welcome.

Live demo

https://video-ragdlproject-hezrklx8rnhbjwucqvegxq.streamlit.app

Contact

Sanjay Reddy Choudapur · Schoudapur1@student.gsu.edu